Programmation
Généralités

1 Notion d'algorithme

- Exemples :
  1. Recette de cuisine
  2. Mode d'emploi d'un appareil
  3. Résolution de $ax^2 + bx + c = 0$

Un *algorithme* est une procédure permettant de résoudre une classe de problèmes, écrite de façon suffisamment détaillée pour pouvoir être suivie sans posséder de compétences particulières ni même comprendre le problème que l'on est en train de résoudre.
Un peu de culture : D'où vient ce mot ? Al-Khawarizmi (né en 783 en Ouzbekistan, mort en 850 à Bagdad), « le père de l'algèbre ».

L'activité de l'informaticien

1. Analyser le problème
2. Modéliser le problème :
   – choisir des structures adaptées à la représentation des données
   – concevoir une stratégie de traitement
   Définition 1 ***Algorithme :*** *ensemble des étapes permettant d'atteindre un but en répétant un nombre fini de fois un nombre fini d'instructions.*
3. coder le problème en traduisant l'algorithme dans un langage donné.

   Définition 2 ***Programme :*** *traduction d'un algorithme en langage compilable ou interprétable par l'ordinateur.*

Un peu plus théorique

Définition 3 *Un algorithme est une méthode générale pour résoudre un ensemble de problèmes.*
– *Il est dit correct lorsque, pour chaque instance du problème, il se termine en produisant la bonne sortie, c'est-à-dire qu'il résout le problème posé.*
– *On mesure l'efficacité d'un algorithme notamment par sa durée de calcul, en partant du principe que chaque instruction a un temps d'exécution constant, par la précision des résultats obtenus, etc.*
– *L'analyse de la complexité algorithmique permet de prédire l'évolution en temps calcul nécessaire pour amener un algorithme à son terme, en fonction de la quantité de données à traiter.*

Donald Knuth (1938 -), professeur à l'université de Stanford, lista les propriétés suivantes comme étant les prérequis d'un algorithme :
– La finitude : un algorithme doit se terminer après un nombre fini d'étapes
– définition précise : «Chaque étape d'un algorithme doit être définie précisément, les actions à transposer doivent être spécifiées rigoureusement et sans ambiguïté pour chaque cas.
– rendement : toutes les opérations que l'algorithme doit accomplir doivent être suffisamment basiques pour pouvoir être en principe réalisées dans une durée finie par une personne utilisant un papier et un crayon.
Il y a des méthodes qui permettent de vérifier qu'un algorithme fait ce que l'on veut.

2 Notion de programme

Un *programme* est la traduction d'un algorithme dans un langage $\begin{cases} \text{compréhensible par l'homme} \\ \text{interprétable par une machine} \end{cases}$

Il est constitué d'un assemblage d'instructions regroupées dans un fichier-texte appelé *code-source* ou *script*.

Programme écrit script	$\Longrightarrow$	Interpréteur Python – Vérification – traduction en langage Assembleur	$\Longrightarrow$	Exécution des commandes

L'exécution d'un programme
– commence à la première instruction
– continue en exécutant d'autres instructions en suivant des règles précises.
- *flot d'exécution :* Le parcours des instructions au cours de l'exécution
- *l'état de l'exécution du programme à l'instant t ;* L'ensemble des variables définies à un instant donné t de l'exécution d'un programme

3 Les instructions d'un programme

Dans un programme, une *instruction* est un ordre de modification de l'état courant de l'exécution d'un programme.
Deux grandes familles d'instructions :

1. Les *instructions simples* qui manipulent directement l'état courant : déclarations et affectations.
2. les *instructions composées* qui permettent d'assembler d'autres instructions et de modifier le flot d'instructions en fonction de l'état courant : Il y en a trois
   (a) la *séquence* qui permet d'exécuter deux instructions l'une à la suite de l'autre.
   (b) le *test* ou *instruction conditionnelle* qui permet de n'exécuter une instruction que dans certains cas.
   (c) la *boucle* qui permet d'exécuter plusieurs fois la même instruction dans un programme.

Théorème 1 *Théorème de Church-Turing : tout procédé de calcul pouvant être décrit de façon systématique peut l'être avec ces cinq instructions*

4 Exemple

```
a=input('entrer le germe: ')
n=input('indice: ')
k=0
u=a
while k<=n:
    if u%2==0:
        u= u//2
    else:
        u= 3*u+1
    k=k+1
print(u)
```

1. Quelles variables et quel type ?

2. Repérer les instructions

3. On donne $a = 25$ et $n = 6$, donner l'état des variables au cours de la réalisation du programme

4. Documenter le programme

5 Pratique

1. **Documenter un programme**
   Il est indispensable que le programme soit lisible
   – pour pouvoir le reprendre plus tard
   – pour qu'il puisse être compris par une tierce personne.
   Les manières de rendre un programme lisible :
   (a) choisir des noms de variables parlants

(b) ajouter des commentaires qui expliquent le rôle des différentes parties d'un programme.
– En Python : une ligne de commentaire commence par le symbole #

2. **Mettre un programme au point en le testant**
Vérifier si un programme ne produit pas d'erreur au cours de son exécution et qu'il effectue bien la tâche que l'on attend de lui.
Pour cela, on le teste : on exécute plusieurs fois le programme en lui fournissant des entrées, appelées *tests* qui permettent de détecter les erreurs éventuelles.
On choisit ces valeurs de tests pour que
(a)
(b)
(c) les cas-limites

Il ne suffit pas que le programme renvoie une valeur pour que le programme soit juste.
Il ne suffit pas que le programme marche une fois pour qu'il soit juste.

La suite de Syracuse Pour les 10^6 premiers entiers pris comme germe, repérer s'il y en a pour lesquels la suite n'atteint pas 1.